Comments on a Variable Guidance Cycle as a LM TLOSS Fix

Reference: Revision 3 of AGC Program Zerlina by Zoroaster, dated 1 March 1970, received via direct mail from G&N contractor 16 March 1970.

Early in the development of the LM guidance program for descent, it was recognized that a major constraint would be the execution time required to accomplish all the functions necessary (guidance, special displays with high update requirements, and elaborate autopilot computations for control of the descent engine gimbal). By careful review of the coding, including substitution of machine language for interpretive language computations where practical, the G mission program was delivered with an execution time margin that probably was in excess of 10%. Due to an RR CDU interface situation, however, an unanticipated loss of computational capacity, amounting to about 15%, was experienced, leading to a computer "overload" situation that resulted in several software restarts. Post-flight study identified the cause of the problem, and also led to a consideration of various ways in which the "TLOSS" phenomenon could be simulated, and software techniques whereby its impact could be minimized. One technique that was recognized was to have an "adaptive" guidance cycle, in which the next guidance computation would not be started until the previous one was definitely completed, as contrasted with forcing such cycles to be initiated every two seconds, with their attendant possibilities for piling up unfinished computations. Since the program tag for the start of the two-second job computations is "SERVICER", this approach became known as the "variable SERVICER" technique.

Tests before the H1 flight revealed what seemed to be an adequate timing margin, however, and in addition it was realized that implementation of a variable SERVICER would have effects on many areas of the program that were written under the tacit assumption of a two-second cycle (imbedded in a number of fixed-memory constants such as those for time-to-go, for example). In addition, there was concern about the effects on downlink monitoring procedures of separating the two-second telemetry word list cycle from the guidance period. Consequently, no authorization was provided to change the two-second SERVICER cycle, and in fact at ASSCCB meeting #33 (13 November 1969) it was explicitly decided that a variable SERVICER cycle should not be included in the "block change" LM package then being considered (which featured the since-rejected Delta guidance).

Interest in TLOSS was revived as the result of some tests made on a modification to the originally released LM H2 program that was made to allow "auto nulling" of velocity errors during the final portion of descent. These tests revealed that at a comparatively low (about 8%) value of TLOSS it was possible for throttle computations to be phased in such a way as to give undesirable excursions in the setting of the throttle. As a result, another modification of the program was released that featured a "load shedding" strategy, whereby attempts to determine the impending buildup of unfinished computations in the final-phase program (P66) were made in the coding, and if encountered then a premature termination of the current guidance cycle is made. Without a work-around, however, potential problems can still be encountered with "lurking" computations from earlier in descent, and hence the H2 program is recognized as being less than an ultimate solution. A general strategy of "graceful degradation" as contrasted with abrupt "cliffs" in the performance of the software has been enunciated, and hence interest in the variable SERVICER technique has again been awakened.

The referenced program was received with the acknowledgment that "many of the little modifications to other programs to make them work with the new variable SERVICER haven't been made yet." Nevertheless, the general strategy to be used is evident, and except for transition to abort programs from descent (or P70 to P71) seems to provide a suitable framework for an eventual flight-qualified program. Some of the salient features of the coding in the reference are:

    1. A lower limit of two seconds on the guidance cycle is imposed (if get done in less than two seconds, delay for the remaining time). This avoids a complete loss of a guidance cycle's data on telemetry (barring some unfortunate phasing relationships which can occur with the present system too).

    2. There is no explicit upper limit, although a program alarm is generated if time since the previous evaluation has exceeded four seconds. Overflow = 10.24 sec.

    3. Accelerometers are not reset when they are read, but instead the accelerometer "output" for the previous evaluation interval is determined by first differencing. Accelerometers are read as part of a job, with interrupts inhibited: this is a good solution to the problem of how restarts in this area are handled, as well as making somewhat more convenient other computations. Accelerometer overflow handled by special coding.

    4. A new sampling of the landing radar (2 V, H, 3 V as authorized by PCR 896) is not initiated until the previous sample has been incorporated into the state vector. This means, of course, that the computations deleted by PCR 896 must be restored to the coding, in order to update the state vector to the LR sampling time before computing the LR/LGC differences for incorporation. This seems unavoidable, but is an illustration of the sort of drawback the variable SERVICER concept introduces: by allowing the computations to exceed the previous maximum of two seconds, the _minimum_ cycle time is also increased.

    5. The complete guidance cycle runs at priority 20_8 as part of a "continuous job" (like "ATERJOB" in CMC during boost for H2, for example), with the minimum of two seconds forced by "DELAYJOB" entrance. This means that the normal "end of job" terminations currently in the programs must be replaced by returns to the check for minimum time. This also means that no "in-line" display computations can occur. Instead, the planned strategy seems to be one of increasing priority temporarily to 23_8, using an "R" type display interface routine, and then returning priority to 20_8 (the "PRIOCHNG" logic will cause the 23_8 job to be executed when it is entered to return priority to 20_8, of course).

    6. The intent seems to be to handle restarts in the prime guidance loop by entrances to the "SERVCHNG" routine, a fixed-fixed memory subroutine that sets up an erasable memory restart in group 5. An analogous procedure is followed in the H2 program, except using group 3, by the "FASTCHNG" routine during the descent computations. LR reading not protected.

    7. The R10/R11 "service" task (R09 in Section 4 GSOP) is done every $\frac{1}{4}$ second continuously, while P66 is executed when reached as part of the navigation sequence and again one second later (by waitlist call), at least per the program in the reference. The second execution is priority 22_8. Presumably the overlapping problem seen in the 3rd H2 release is avoided somehow. P66 changes incomplete in reference.

The following areas will need consideration prior to release of a formal flight program using a variable SERVICER. These are in addition to such external items as management authorization procedures and downlink monitoring impacts. No attempt was made to review the detailed coding of the reference, since it was not advertised to be the final version and contained a number of assembler alarms (mostly undefined tags).

1. Validity of assumptions. For example, the accelerometer first differencing scheme (item 3 above), which uses a check of bit 14 of the first difference to determine the sign of the result, will give an output with the wrong sign if an accelerometer has incremented by more than 81.91 meters/second since the previous sample. This fact is duly noted in the comments field of the listing, but in view of the LM-1 difficulty with an "obscure" part of the program, namely Delta-V monitor constant value, such items deserve and require a formalized enumeration and an active MSC approval. P66 ROD interactions presumably satisfactory now, but should be checked.

2. As mentioned previously, the program of the reference does not seem to accommodate transitions to abort programs as satisfactorily as did the H2 program. The prime reason for this is that guidance is restart protected with the "Average-G" restart group of 5 in the reference, whereas in the H2 program it was group 3, meaning that abort/abort stage inputs, which disabled group 3, would halt the descent coding appropriately if the program happened to have reached that area. The present P70/P71 technique of reducing priority to 17_8, in fact, before "connecting" ascent coding could cause a considerable delay if the existing cycle was running over 2 seconds. Changes to the ascent coding are incomplete, however (restart group 5.3 is used at the start, for example, which is an undefined tag reference).

3. There are a number of imbedded constants in the program that reflect the two-second cycle, some of which presumably need to be changed. For example, "S40.8" (time-to-go for P40 and P42) uses the computing interval of 2 sec. in its formula (see page 5.3-35 of Section 5 Rev. 7 Luminary GSOP). In addition, a criterion of 4 seconds (since accelerometer reading) is used to decide whether to terminate guidance and initiate a waitlist call for engine off, and this number presumably needs reconsideration. Furthermore, there are implicit ("K:dvtoacc" for OMEGAPD on page BURN-22 of official MSC Luminary guidance equations) and explicit ("K:DTdDELT" for DELCDU, same page) constants in "FINDCDUW" that are based on a two-second cycle. Note that if a new value is not provided from "FINDCDUW", the previous DAP interface cells are untouched, and hence the rate is maintained, not an "attitude hold".

4. In several cases the program is oriented towards a two-second update cycle: probably in some of these any sloppiness in the interval can be demonstrated to be benign; in other cases there may have to be special provisions made or analyses conducted. For example, the ascent thrust filter (page 5.3-144 of Section 5 Rev. 7 Luminary GSOP) assumes in steady-state that four accelerometer outputs, each for a two-second interval, have been obtained; the delta-V monitor constants are in units of velocity whereas the actual quantity to be measured is thrust (see PCR 612.2).

The reference is a good start towards a flight-qualified variable SERVICER set of coding. A variety of problems, however, in many segments of the program need to be identified and resolved correctly before the program is used on a flight. The variable SERVICER has much appeal as an "ultimate" fix to the TLOSS phenomenon, however, and assuming a continuation of Apollo launches probably should be implemented.